|  |  |
| --- | --- |
| Borland Delphi OEP | Borland C++ OEP |
| PUSH EBP MOV EBP,ESP ADD ESP,-x (which x is a number) MOV EAX, adr1 CALL adr2 ... ....  adr1: (adr1 is the address of secons ??s) C3 E9 ?? ?? ?? FF EB F8 5D C3 ?? ?? 00 00  adr2: (adr2 is start of *InitExe* routine which has a call to GetModuleHandleA)  InitExe:... ...  PUSH 0  CALL <JMP.&kernel32.GetModuleHandleA> ... ...  Bytes to binary paste in OllyDbg (11 should be replaced)  55 8B EC 83 C4 11 B8 11 11 11 11 E8 11 11 11 11 | JMP Label 01  BOUND DI,DWORD PTR DS:[EDX]  INC EBX  SUB EBP,DWORD PTR DS:[EBX]  DEC EAX  DEC EDI  DEC EDI  DEC EBX  NOP  JMP 00AA26C2 Label 01:  MOV EAX,DWORD PTR DS:[adr]  SHL EAX,2  MOV DWORD PTR DS:[adr + 0x4],EAX  PUSH EDX  PUSH 0  CALL <JMP.&KERNEL32.GetModuleHandleA>  MOV EDX,EAX  CALL \_\_\_CRTL\_VCL\_Init   POP EDX  CALL NULL\_sub ;*NULL\_sub* is a function with just a RET. This Line can be NOPed too.  CALL NULL\_sub  PUSH 0  CALL \_\_ExceptInit  POP ECX  PUSH adr – 0x57   PUSH 0  CALL <JMP.&KERNEL32.GetModuleHandleA>  MOV DWORD PTR DS:[ adr + 0x8],EAX  PUSH 0  JMP \_\_startup  JMP \_\_GetExceptDLLinfoInternal  XOR EAX,EAX  MOV AL,BYTE PTR DS:[adr – 0xE]  RETN  MOV EAX,DWORD PTR DS:[adr + 0x8]  RETN  **Junk code**  =====================================  adr can be calculated easily by the other lines. For example if the last line is:  MOV EAX,DWORD PTR DS:[401248]  RETN  Then adr = 401248 – 8 = 401240  ===============================  Bytes to binary paste in OllyDbg (11 should be replaced)  EB 10 66 62 3A 43 2B 2B 48 4F 4F 4B 90 E9 28 01 4B 00 A1 11 11 11 11 C1 E0 02 A3 11 11 11 11 52 6A 00 E8 11 11 11 11 8B D0 E8 11 11 11 11 5A E8 11 11 11 11 E8 11 11 11 11 6A 00 E8 11 11 11 11 59 68 C4 00 4B 00 6A 00 E8 11 11 11 11 A3 11 11 11 11 6A 00 E9 11 11 11 11 E9 11 11 11 11 33 C0 A0 11 11 11 11 C3 A1 11 11 11 11 C3 |